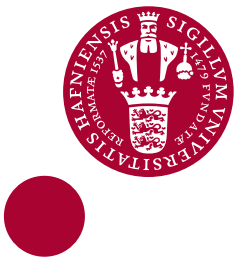

Learning models of activities involving interacting objects

Manfredotti, Cristina; Pedersen, Kim Steenstrup; Hamilton, Howard J.; Zilles, Sandra

Published in:
Advances in Intelligent Data Analysis XII

DOI:
10.1007/978-3-642-41398-8_25

Publication date:
2013

Document version
Peer reviewed version

Learning Models of Activities Involving Interacting Objects

Cristina Manfredotti[1], Kim S. Pedersen[2], Howard Hamilton[3], Sandra Zilles[3]

[1] LIP6, Pierre and Marie Curie University (UPMC), Paris, France
[2] DIKU, University of Copenhagen, Copenhagen, Denmark
[3] Department of Computer Science, University of Regina, Regina, Canada

Abstract. We propose the LEMAIO multi-layer framework, which makes use of hierarchical abstraction to learn models for activities involving multiple interacting objects from time sequences of data concerning the individual objects. Experiments in the sea navigation domain yielded learned models that were then successfully applied to activity recognition, activity simulation and multi-target tracking. Our method compares favourably with respect to previously reported results using Hidden Markov Models and Relational Particle Filtering.

1 INTRODUCTION

Many practical problems including activity recognition, multi-target tracking and detection of activity, require reasoning about the interactions of multiple related objects. A complex activity (such as exchanging goods between two ships) is a type of interaction usually realized as a sequence of lower-level actions that may involve multiple objects. Given a model of how an activity is decomposed, it is possible to effectively recognize an ongoing activity by observing low-level attributes (such as position, speed and color) [3, 17]. However, the model for such activities is usually unknown. We investigate one possible solution: learning such a model directly from sensor data. This paper introduces a framework, called LEMAIO (LEarning Models of Activities involving Interacting Objects), for learning probabilistic models of complex activities involving multiple interacting objects from sensor data. This framework is capable of inferring the interactions between the objects, while also inferring how complex activities decompose into lower level actions.

An *activity* is usually recognized from the sequence of the attribute values of the interacting objects. An example of an activity in the soccer domain is "passing the ball", which can be recognized observing the sequence of positions of the players and the ball over time. The same activity can be undertaken in many ways, represented as different sequences of attribute values, e.g., all the ways in which the ball can be passed from one player to another. To avoid listing all possible realizations of an activity, we need an abstract representation (a *model*) of it. This model should be such that an automatic system can use it efficiently to recognize an activity from (noisy) data (*activity recognition*), simulate it (*activity generation*), or track it (*multi-target tracking*).

Since activities often involve multiple objects, modeling the relations between them is crucial for capturing their behaviour. Consider the difference between the activities of "passing" and "intercepting" a ball: both activities result in a new player having control of the ball but the former requires the two players to be on the same team (to be in the relation of having the same value for the team attribute), while the latter requires them to be on different teams. We distinguish between *atomic activities* (called simply "activities" in [1]), involving coordinated actions among multiple agents at one time, and *complex activities*, which are sequences of atomic activities. We do not assume we know the relations between objects: we know that objects might interact, but we do not know how. We assume all relations are pairwise.

We are given a training set where each instance is a sequence of attribute values describing the complete state of the world, along with a label identifying the complex activity represented and we adopt a probabilistic viewpoint: the problem is to learn from this training set a probabilistic model able to identify complex activities in new data, track individual objects while complex activities are occurring, and generate sequences of synthetic data simulating complex activities. The LEMAIO framework addresses this problem by learning a three-layer hierarchical model from the bottom up that can be mapped into a Dynamic Bayesian Network.

The main contributions of our work are: (1) a general top-supervised learning framework to learn a hierarchical probabilistic model for complex activities from low-level data; (2) the decomposition of complex activities into lower level actions and interactions between objects and the explicit modelling of objects' interactions; (3) an implementation of the framework based on Expectation Maximization and clustering; (4) empirical evidence of the effectiveness of our approach in learning models able to recognize, track, and generate complex activities.

2 THE LEMAIO FRAMEWORK

To describe the LEMAIO framework, we first explain the four levels of abstraction. Then we describe how a three-layered model is learned that allows values at any level to be generalized to the next higher level. Finally, we show how the model is used to generate synthetic data corresponding to specified activities.

Our learning approach is *top-supervised*: labels are available in the training data only at the top (complex activity) level. The learned model is able to assign a label to a complex activity represented by an unseen sequence of low-level data and is also able to recognize the lower-level constituents of the activity.

2.1 Levels of Abstraction in LEMAIO

The LEMAIO framework uses four levels of abstraction: (0) attribute values for objects (raw data), (1) single object activities (activities involving only one object), relations between pairs of objects at a single time, and changes in relations over time, (2) atomic activities and (3) complex activities. We assume

that during any time interval an object can be involved in at most one single object activity and a set of related objects can only be involved in at most one atomic activity and at most one complex activity. In this section, we consider each level in turn. These levels of abstraction are general and the decomposition of a complex activity into atomic activities, relations, changes of relations and single object activities can be applied to a variety of domains.

Level 0: We collect all attribute values of the objects in the world at consecutive time steps while some complex activity is occurring. We assume the attribute values correspond to noiseless observations from sensors that coincide with the actual state of the world (in presence of noisy observation we could filter the data before learning). Let $s_t^{(i)}$ be the *state of the object* $o^{(i)}$ at time t. The *state of the world* s_t can be represented as the vector of the states of all individual objects in the world at time t . The training data consists of pairs of sequences of consecutive states of the world from time 1 to time T and labels of the complex activity represented by the sequence $(s_{1:T},\ \gamma)$. Assuming that we have labels for complex activities is restrictive but fair: it is easier for a person to classify a complex activity (providing a labeled instance) than to describe such an activity.

Level 1: At level 1 we represent how objects behave individually, how they interact and how their interactions change over time.

- *single object activities* are associated with changes over time in the attribute values of single objects. $e_{(t-1,t)}^{(i)}$ represents the single object activity that $o^{(i)}$ performs in time interval $(t–1, t)$ and assumes values in $\mathcal{E} = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_{n_E}\}$.
- *relations* represent degrees of similarity between attribute values of objects at the same time. We assume pairwise relations. $r_t^{(i,j)}$ represents the relation between $o^{(i)}$ and $o^{(j)}$ and assumes values in $\mathcal{R} = \{\rho_1, \rho_2, \cdots, \rho_{n_R}\}$.
- *changes in relation* represent changes in the degree of similarity over time. $d_{(t-1,t)}^{(i,j)}$ represents the change of relation between $o^{(i)}$ and $o^{(j)}$ during the time interval $(t-1, t)$ and assumes values in $\mathcal{D}=\{\delta_1, \delta_2, \cdots, \delta_{n_D}\}$.

From data classified into single object activities, relations and change in the relations we can learn distributions for atomic activities.

Level 2: Atomic activities describe how one object behaves with respect to another during the time interval between consecutive time points. $a_t^{(i,j)}$ represents an atomic activity involving the related objects $o^{(i)}$ and $o^{(j)}$ in the interval $(t–1, t)$ and it is learned from vectors of the form $a_t^{(i,j)} = [e_{(t-1,t)}^{(i)}, e_{(t-1,t)}^{(j)}, r_t^{(i,j)}, d_{(t-1,t)}^{(i,j)}]$. We define the set of all possible atomic activities as $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_{n_A}\}$.

Level 3: A *complex activity* $c^{(i,j)}$ is represented by a sequence of atomic activities that involve $o^{(i)}$ and $o^{(j)}$; $c^{(i,j)}$ can assume values in $\mathcal{C} = \{\gamma_1, \gamma_2, \cdots, \gamma_{n_C}\}$.

Example: Let us focus on two possible complex activities (*Rendezvous* and *Avoidance*) that can occur at sea. They consist of two vessels approaching each other and subsequently going apart. In the latter only one of the two vessels changes its speed to avoid the other ship, but in the former, when the vessels are close to each other, they stay close with speed near zero to illegally exchange goods. The state of the world is indicated by the position, name and class of

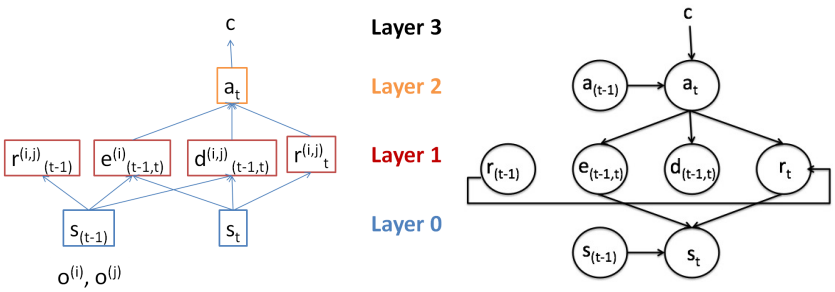

Fig. 1. left: The LEMAIO hierarchy with layers. **right:** The Dynamic Bayesian Network learned by LEMAIO.

each vessel. The training set consists of pairs of a sequence of states and the label (R or A) of the complex activity. A single object activity can encode the movement of a ship (e.g., moving fast towards north); a relation can give the separation distance between two ships or if they are of different/equal type; a change in relation can tell whether the distance between two ships is increasing or decreasing during a time interval; and an atomic activity can describe the idea of "approaching" (e.g., two ships have decreasing distance over time). A complex activity modeling a *Rendezvous* could be composed as a sequence of atomic activities such as "approaching", "staying together" and "going apart" (each possibly repeated).

2.2 Learning with LEMAIO

In order to model the uncertainty about which activities are currently being performed and how a complex activity decomposes into lower-level constituents, we introduce a number of probability distributions. These are models that, given an observed pattern and using the Bayes theorem, (1) assign probabilities to the events that associate the pattern with any (single object, atomic, or complex) activity and (2) assign probabilities to the future. In this way, the learned model can be used for classification and generative purposes. We separate our presentation into levels based on the abstraction hierarchy shown in Fig. 1 left.
Single Object Activities and Relations: To learn models for these quantities we preprocess the data into three sets of differences.

- $\Delta^{(i)}_{(t-1,t)}$ denotes the difference between the states of $o^{(i)}$ at two consecutive time points: $\Delta^{(i)}_{(t-1,t)} = s^{(i)}_t - s^{(i)}_{t-1}$, where $t > 0$;
- $\Delta^{(i,j)}_t$ denotes the distance between the states of $o^{(i)}$ and $o^{(j)}$ at the same time step: $\Delta^{(i,j)}_t = dist(s^{(i)}_t, s^{(j)}_t)$, where $t \geq 0$, $i \neq j$;
- $\Delta^{(i,j)}_{(t-1,t)}$ is short for: $\Delta^{(i,j)}_{(t-1,t)} = \Delta^{(i,j)}_t - \Delta^{(i,j)}_{t-1}$,where $t > 0$, $i \neq j$.

Since single object activities involve only one object, they can be seen as the change in the attribute values for one object. Given a sequence of states (our training data), for all the attribute values of every object of every pair of consecutive time steps we compute $\Delta^{(i)}_{(t-1,t)}$. From these data, we learn a model for single object activities. The model consists of the prior of the single object activity class, $p(e^{(i)}_{(t-1,t)}=\epsilon_k)$, and the probability density function $p(\Delta^{(i)}_{(t-1,t)}|e^{(i)}_{(t-1,t)}=\epsilon_k)$.

A relation is a difference between the attribute values of two objects at a single time point. We learn a probability distribution for relations from data of the form $\Delta^{(i,j)}_t$ obtained from the states of every pair of objects in the training data at the same time step. The model consists of the prior of the relation class, $p(r^{(i,j)}_t=\rho_k)$, and the probability density function $p(\Delta^{(i,j)}_t|r^{(i,j)}_t=\rho_k)$.

A change in relation is a difference between the relation of two objects over time. From $\Delta^{(i,j)}_{(t-1,t)}$, we learn the prior of the classes of changes in relations, $p(d^{(i,j)}_{(t-1,t)}=\delta_k)$, and the probability density function $p(\Delta^{(i,j)}_{(t-1,t)}|d^{(i,j)}_{(t-1,t)}=\delta_k)$.

According to the Bayes formula the probability of a single object activity class ϵ_k, given the observed data $\Delta^{(i)}_{(t-1,t)}$, is the posterior

$$p(e^{(i)}_{(t-1,t)}=\epsilon_k|\Delta^{(i)}_{(t-1,t)})=\frac{p(e^{(i)}_{(t-1,t)}=\epsilon_k)p(\Delta^{(i)}_{(t-1,t)}|e^{(i)}_{(t-1,t)}=\epsilon_k)}{p(\Delta^{(i)}_{(t-1,t)})}. \quad (1)$$

Such a posterior can be used for classification purposes. Similar posteriors can be derived for relations and changes in relations.

Atomic Activities: To learn a model for atomic activities we first apply the probabilistic models learned at layer 1 (Eq. 1) to classify data into single object activities, relations and changes of relations. Secondly, by considering every time interval in every input sequence, we collect vectors $v^{(i,j)}$ of single object activities, relations and changes in relations for every pair of related objects:

$$v^{(i,j)}=[e^{(i)}_{(t-1,t)}, e^{(j)}_{(t-1,t)}, r^{(i,j)}_t, d^{(i,j)}_{(t-1,t)}]. \quad (2)$$

We then cluster these vectors for all (i, j) pairs according to a distance measure f. Next, for each cluster (α_k) we select one vector (v_{α_k}) to represent all vectors in the cluster. Finally we map each cluster into the set of labels in $\mathcal{A}$. We assume the probability that a vector $v^{(i,j)}$ is in cluster α_k, $p(v^{(i,j)}|a^{(i,j)}=\alpha_k)$, is given by one minus its normalized distance from v_{α_k}: $p(v^{(i,j)}|a^{(i,j)}=\alpha_k)=1-\tilde{f}(v^{(i,j)}, v_{\alpha_k})$. We learn the prior $p(a^{(i,j)}=\alpha_k)$ proportional to the number of the data points in the training set that fall in cluster α_k. To classify vectors of the form of Eq. 2 into atomic activities we can use the posterior

$$p(a^{(i,j)}=\alpha_k|v^{(i,j)}) \propto p(a^{(i,j)}=\alpha_k)p(v^{(i,j)}|a^{(i,j)}=\alpha_k).$$

Complex Activities: Complex activities are defined as sequences of atomic activities. We group the data in the training set according to their complex activity label and, using the distributions learned at the previous layers, we map

them into sequences of atomic activities. From these sequences we learn the probability that an atomic activity α_k follows a sequence of atomic activities $a_{1:t-1}^{(i,j)}$ given a particular complex activity γ_k:

$$p(a_t^{(i,j)} = \alpha_k | c = \gamma_k, a_{1:t-1}^{(i,j)}). \tag{3}$$

The probability $p(a_1 = \alpha_k)$ is proportional to the number of times it occurs at time $t=1$ in the training set. The prior $p(c=\gamma_k)$ is proportional to the number of occurrences of γ_k in the training set. We classify a sequence of atomic activities $a_{1:t}^{(i,j)} = \{a_1^{(i,j)}, a_2^{(i,j)}, \cdots, a_t^{(i,j)}\}$ as the complex activity $c=\gamma_k$ that is associated with the highest value of $p(c=\gamma_k | a_{1:t}^{(i,j)})$, where

$$p(c = \gamma_k | a_{1:t}^{(i,j)}) = \frac{p(a_t^{(i,j)} | c = \gamma_k, a_{1:t-1}^{(i,j)}) p(a_{1:t-1}^{(i,j)}, c = \gamma_k)}{p(a_{1:t}^{(i,j)})}. \tag{4}$$

The overall learned model is depicted in Fig. 1 right.

2.3 Activity Generation with LEMAIO

Activity generation aims at generating sequences of states that match a given complex activity $c = \gamma_k$. Assume we are given a sequence of states $s_{0:t-1}$ that matches complex activity γ_k. Given the probability distributions learned so far, $s_{0:t-1}$ can be classified into sequences of atomic activities $a_{1:t-1}^{(i,j)}$. We want to generate the next atomic activity $a_t^{(i,j)}$ such that the sequence $a_{1:t}^{(i,j)}$ is constrained to be associated with complex activity $c = \gamma_k$. To do so, we sample from the probability distribution $p(a_t^{(i,j)} | c = \gamma_k, a_{1:t-1}^{(i,j)})$ learned at layer 3. Suppose $a_t^{(i,j)} = \alpha_k$. Next, we sample a vector $v^{(i,j)}$ from the probability distribution $p(v^{(i,j)} | a_t^{(i,j)} = \alpha_k)$ learned at layer 2. Suppose $v^{(i,j)} = [\epsilon_i, \epsilon_j, \rho_l, \delta_m]$, telling us the generated single object activities, relations and changes of relations.

Knowing the current state s_{t-1}, single object activities (ϵ_i and ϵ_j), relations (ρ_l) and change in relations (δ_m) we can generate the next state s_t by sampling from the probability distributions learned at layer 1. To model the change in relation, let us introduce a random variable D_m: $D_m \sim p(\Delta_{(t,t-1)}^{(i,j)} | d_{(t,t-1)}^{(i,j)} = \delta_m)$. Let q be the distribution of the random variable $\Delta_{t-1}^{(i,j)} + D_m$. We can sample a value for $\Delta_t^{(i,j)}$ from the probability

$$p(\Delta_t^{(i,j)} | d_{(t-1,t)}^{(i,j)} = \delta_m, r_t^{(i,j)} = \rho_l, \Delta_{t-1}^{(i,j)}) = q(\Delta_{t-1}^{(i,j)} + D_m) p(\Delta_t^{(i,j)} | r_t^{(i,j)} = \rho_l),$$

estimated from the distributions of the relations and their changes.

To simplify the following explanation, we assume the only objects in the world are $o^{(i)}$ and $o^{(j)}$. Given the sampled value for $\Delta_t^{(i,j)}$, we can sample $s_t = [s_t^{(i)}, s_t^{(j)}]$ from $p(s_t | \epsilon_i, \epsilon_j, s_{t-1}, \Delta_t^{(i,j)})$. That, assuming $p(s_t^{(i)} | s_{t-1}^{(i)}, \Delta_t^{(i,j)}) = p(s_t^{(i)} | s_{t-1}^{(i)})$, can be factored as:

$$p(s_t | \epsilon_i, \epsilon_j, s_{t-1}, \Delta_t^{(i,j)}) = \frac{p(s_t^{(i)} | \epsilon_i, s_{t-1}^{(i)}) p(s_t^{(j)} | \epsilon_j, s_{t-1}^{(j)}) p(s_t^{(j)} | s_{t-1}^{(j)}, \Delta_t^{(i,j)}, s_t^{(i)})}{p(s_t^{(j)} | s_{t-1}^{(j)})}. \tag{5}$$

With this assumption, we can first sample the state of $o^{(i)}$ and then sample the state of $o^{(j)}$ taking into account the state of $o^{(i)}$ already sampled and their relations. This assumption is equivalent to assuming one of the two objects ($o^{(i)}$ in this case) is the "leader" and can be loosened in practice by exchanging the order in which the objects are processed at each time step.

2.4 An Implementation

In our LEMAIO-1 implementation of the LEMAIO framework, we use mixtures of Gaussians, for the distributions at layer 1, a mixture of categorical distributions at layer 2 and a mixture of Markov chains at layer 3.

LEMAIO-1 Layer 1: Since the same approach is used for the three kinds of entities at layer 1, here we present only the procedure for learning single object activities. We use Expectation Maximization (EM) to learn the prior of the classes of single object activities and the probability density function of the data given the class that maximize the likelihood of the data [6]. Assuming we have observed a particular change in one object's attribute values ($\Delta^{(i)}_{(t-1,t)}$), the likelihood of the observed data given the parameters Θ of the distributions is calculated as: $p(\Delta^{(i)}_{(t-1,t)}|\Theta) = \sum_{k=1}^{K} p(\epsilon_k)p(\Delta^{(i)}_{(t-1,t)}|\epsilon_k, \Theta)$ where K is the number of classes represented in the data, chosen to minimize the Bayes Information Criterion (BIC) [7], and Θ is the vector of the means and variances of the chosen Gaussian distributions.

LEMAIO-1 Layer 2: To cluster vectors $v^{(i,j)}$ (cf. Eq. 2) into atomic activity classes we use the K-medoids clustering algorithm [14]. We compute the distance measure $f(v_1, v_2)$ on which the clustering is based in the following way: first we map each element in v_1 and in v_2 into the mean of the Gaussian distribution that best fits the class represented by the element; then we compute the Euclidean distance between these elements and average over the elements of the vectors. The number of clusters K is chosen such that it maximizes the intracluster similarity of our data. The number of distributions and the number of clusters K is chosen given a maximum number of sets $\mathcal{K}$.

LEMAIO-1 Layer 3: Given sequences of atomic activities labeled with the same complex activity ($c = \gamma_k$), we learn a Markov chain. In a Markov chain, the probability of an atomic activity at time step t depends only on the value of the atomic activity at time step $t-1$ and on the current complex activity $c=\gamma_k$. We thus have (Eq. 3): $p(a_t^{(i,j)} = \alpha_k | c = \gamma_k, a_{1:t-1}^{(i,j)}) = p(a_t^{(i,j)} = \alpha_k | c = \gamma_k, a_{t-1}^{(i,j)})$. Modeling the transitions between atomic activities with a Markov chain allows us to simplify the classification of complex activities writing Eq. 4 as:

$$p(c = \gamma_k | a_{1:t}^{(i,j)}) = p(c = \gamma_k) \prod_{t=2}^{T} p(a_t^{(i,j)} = \alpha_k | c = \gamma_k, a_{t-1}^{(i,j)}). \quad (6)$$

3 EXPERIMENTS

We experimented with our implementation on the sea navigation data set provided in [4]. This data set is composed of 37 sequences called *encounters*. An

encounter is a sequence of 96 time steps recording the 2D positions of two ships involved in either a rendezvous or an avoidance; there are 19 rendezvous encounters. In the following, we describe how we applied LEMAIO-1 to learn models from this dataset and how we tested the resulting models.

In our data set the state $s_t^{(i)}$ is the vector $[x_t^i, y_t^i]$ of the position of $o^{(i)}$. As distance between $s_t^{(i)}$ and $s_t^{(j)}$ we use the Euclidean distance. In this way we learn the following probabilistic models: i) for single object activities from vectors representing the *movement* of individual objects, ii) for relations from *distances* between objects and iii) for changes in relations from *differences of distances during time*. Given the small number of encounters in the data set, we adopted the *leave-one-out* cross-validation technique [10].

At the first level, to choose K, we set $\mathcal{K}1$ to 10 and apply BIC. We restrict the EM algorithm to iterate for a maximum of 1000 times and add a small regularization factor $(1e-5)$ to the diagonal of the covariance matrices to ensure they are positive-definite. For the K-medoid algorithm, we fix $\mathcal{K}2$ to 20 and the maximum number of iterations to 500. On average, the number of Gaussians learned for single object activities is 9, for relations 7 and for changes of relations 9. On average, the number of clusters the K-medoid algorithm finds is 19. Similar results were obtained with different values of $\mathcal{K}$. We learned two Markov chains, one for each type of encounters. To avoid having transitions of probability 0 for unobserved patterns, we used *Laplace's succession rule*.

We tested the models learned by LEMAIO-1 on activity classification, activity generation, and multi-object tracking. Moreover, we tested these models on online activity recognition and compared the results to [17].

Experiment 1: Encounter Classification

Rendezvous (Positive)	19
Avoidance (Negative)	18
True Negative	11
True Positive	19
False Positive	7
False Negative	0
Accuracy	0.81
True Negative Rate	0.61
Recall (True Positive Rate)	1
Precision	0.73
F-measure	0.84

To test an unseen encounter, our classification method assigns it the label (rendezvous or avoidance) associated with the Markov chain with the highest likelihood. Since the data includes 37 encounters, we trained and tested 37 different models: each model was trained on 36 of the 37 encounters and tested on the remaining encounter. The results are reported in the table above. Our method had an F-measure of 0.84. A lower F-measure of 0.72 was previously reported [17] on the same data set using hidden Markov models, obtained with a supervised approach, whereas our system is (only) top-supervised.

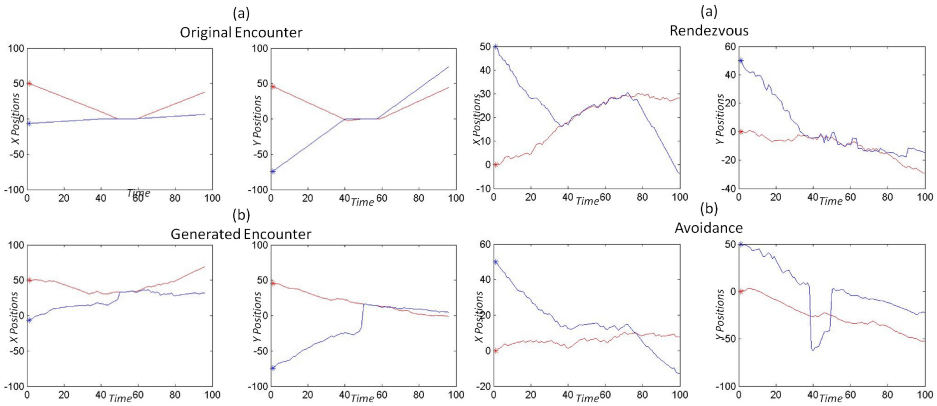

Fig. 2. left: An example encounter generated given the atomic activities: (a) the original encounter from the test set and (b) an encounter generated from the atomic activities recognized from the original encounter. **right:** Two examples of encounters generated given complex activities: (a) a rendezvous and (b) an avoidance.

Experiment 2: Encounter Generation

To evaluate the suitability of the models learned by LEMAIO-1 for generating encounters probabilistically, we ran two experiments of increasing complexity.

In *Experiment 2a* (generation given a sequence of atomic activities), for each learned model, we took the encounter part of the test set, and classified it to give a sequence of atomic activities. From this sequence we generated the low-level data representing an encounter, i.e., we generated the positions of the two ships. One of these encounters is shown in Fig. 2 left, where the original rendezvous from the test set is shown at the top and a rendezvous generated by the learned model is shown at the bottom. Notice that the generated tracks follow the paths of the original ones, as dictated by the recognized atomic activities.

In *Experiment 2b* (generation given a complex activity; by first generating a sequence of atomic activities and then generating encounters from them), for each model we generated a sequence of atomic activities $a_{1:T}$ for the rendezvous complex activity and another sequence for the avoidance activity. We sampled the atomic activity sequence according to the Markov chain learned, by first sampling the first atomic activity a_1 according to the vector of priors in the Markov chain (associated with the relevant encounter) and then sampling the atomic activity a_{t+1} according to the probability of transition from the atomic activity a_t. Fig. 2 on the right represents a rendezvous (top) and an avoidance (bottom) generated from one model learned by LEMAIO-1. In both cases the ships are approaching at the beginning and going apart at the end. In the rendezvous, there is a distinctive behaviour localized in the center where the two ships stay close together for a while; this does not happen in the avoidance.

For both Experiment 2a and 2b the generation of a sequence of positions given a sequence of atomic activities is done with sampling. From each atomic activity in the sequence we sample a particular vector of probabilistic models that gives us the single object activities of $o^{(i)}$ ($e_t^{(i)}$), and of $o^{(j)}$ ($e_t^{(j)}$), their relation $r_t^{(i,j)}$ and the change of their relations $d_t^{(i,j)}$ at time t. For each atomic activity we generate $M_2 = 100$ vectors and from each of these $M_1 = 100$ positions following Eq. 5. We sample M_1 positions of $o^{(i)}$ and $o^{(j)}$ independently from $p(s_t^i|\epsilon_i, s_{t-1}^i)$ and $p(s_t^j|\epsilon_j, s_{t-1}^j)$, resp. We sample M_1 distances $\Delta_t^{(i,j)}$ from Eq. 5 on the line (s_t^i, s_t^j) and, for each sample, we fix one of the sampled $s_t^{(i)}$ or $s_t^{(j)}$ and pick the other at the opposite side at distance $\Delta_t^{(i,j)}$. To avoid preferential treatment, we exchange the order in which $s_t^{(i)}$ or $s_t^{(j)}$ are chosen at each time step.

Experiment 3: Tracking

We evaluate the tracking ability of the models learned by LEMAIO-1. This experiment makes use of the 3PF algorithm presented in [17] coupled for the prediction step with the same transition model used for the generation experiments and learned by our LEMAIO-1 implementation. Each particle first samples the distribution of complex activities, then samples the atomic activities using the appropriate Markov chain, and then predicts the next position of each object based on the atomic activities. Thus, while tracking, the algorithm is also able to recognize the activity online. When an observation arrives the tracker filters it by weighting the particles according to a sensor model that takes into account their distance from the observation. For comparison purposes we used the same sensor model used in [17] and the same number of particles ($M = 100$).

We compare the tracking performance of the 3PF algorithm using the models learned by LEMAIO-1, with the performance of the original 3PF (that uses a model manually optimised for tracking) and a standard particle filtering algorithm (PF)[4]. The mean of the tracking errors on 37 encounters for 3PF using LEMAIO-1 models is 0.27, for the original 3PF is 0.15 and for the standard PF is 1.68. As expected, the tracking error with the LEMAIO-1 models is higher than that obtained with the hand-crafted model, but it was substantially better than the standard PF. The accuracy for the activity recognition task is 0.95.

4 DISCUSSION

The LEMAIO multi-layer framework learns models for activities involving multiple interacting objects from sequences of attribute values for the individual objects. Our experiments show the validity of the models learned on a publicly available data set. In particular, our results are better than previously reported results using Hidden Markov Models (for activity recognition) and Relational Particle Filtering (for tracking).

Numerous researchers have dealt with the problem of modeling and recognizing the actions of a single agent [2, 19]. Single object activities can be used to

[4] The tracking error for an encounter is computed as the mean distance between the filtered and actual positions of the ships across all time points.

recognize the action of an agent in a time interval. In practice, many activities of interest involve several agents, which interact with each other and with the environment. LEMAIO is better suited to such problems than the single agent approaches because it learns a model for the relations between interacting objects and the way these relations change over time, in contrast to other approaches that consider relations between objects by either limiting the interactions to particular types [9] or constraining the objects and their interactions to be fixed over time [8]. Many works have dealt with the problem of representing and recognizing complex activities from data [20–22]. These approaches typically rely on a model of the activity being provided by a domain expert. Such models are rarely available for real life systems featuring many variables with complex interdependencies. As well, many of these models are inflexible and can be used for recognition but not for generation or tracking. In contrast, LEMAIO learns its own model, which can be used for recognition, generation and tracking. Some existing approaches use a hierarchy of actions specified by a stochastic context free grammar [13, 15], making them less flexible than LEMAIO.

Several approaches have dealt with learning concepts similar to the ones learned at the various layers of the LEMAIO framework. For example, the equivalent of single object activities has been learned in various computer vision systems [16]. Atomic activities have been used, for example, to recognize robot actions by various RoboCup competitors [18] taking as given the interpretation of low level attribute values. Bobick [1] distinguished the concepts of *actions*, characterized by simple motion patterns typically executed by a single agent, and *activities*, which are more complex and involve coordinated actions among multiple agents. To the best of our knowledge LEMAIO is one of the first approaches to put these concepts together and is the first approach to learn models for relations.

Other previous approaches studied the behaviour of several objects moving together by considering them as a single entity [5, 11]. Our aim is to model the behaviour of interacting objects pursuing an activity that is permitted to be something other than moving together. For this reason, we chose to learn a model of the relations that is separate from the model of the activities of the single objects. The relations studied in this paper were based on distance or similarity. We hypothesize that the LEMAIO framework can learn models for relations that are not distance or similarity relations, such as "passing an obstacle on the left" or "being on the same team".

In future work, we will investigate the use of different probability distributions. We think that, especially at the second layer, the use of time windows may improve the accuracy of the model. Therefor, we will investigate the use of Temporal Nodes Bayesian Networks [12]. The assumption that relations are pairwise is certainly a limitation, but investigating all possible combinations of related objects while learning is computationally intractable for a large number of objects. We are investigating the use of non-parametric methods to discover which objects are related while performing a particular activity.

References

1. Bobick, A.F.: Movement, activity and action: the role of knowledge in the perception of motion. Phil. Trans. Lond. B **352** (1997) 1257–1265
2. Bobick, A.F., Davis, J.W.: The recognition of human movement using temporal templates. IEEE Trans. Pattern Anal. Mach. Intell. **23**(3) (2001) 257–267
3. Borg, M., Thirde, D., Ferryman, J.M., Fusier, F., Valentin, V., Brémond, F., Thonnat, M.: Video surveillance for aircraft activity monitoring. In: AVSS. (2005) 16–21
4. CAIAC: The CAIAC intelligent systems challenge. http://www.intelligent-systems-challenge.ca/challenge2009/problemDescriptionAndDataset/index.html (2009)
5. Cattelani, L., Manfredotti, C.E., Messina, E.: Multiple object tracking with relations. In: ICPRAM (1). (2012) 459–466
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. B **39**(1) (1977) 1–38
7. Fraley, C., Raftery, A.E.: How many clusters? Which clustering method? Answers via model-based cluster analysis. The Computer Journal **41**(8) (1998) 578–588
8. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: IJCAI. (1999) 1300–1309
9. Galata, A., Cohn, A.G., Magee, D.R., Hogg, D.: Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models. In: ECAI. (2002) 741–745
10. Geisser, S.: Predictive Inference. Taylor & Francis (1993)
11. Gning, A., Mihaylova, L., Maskell, S., Pang, S., Godsill, S.: Group object structure and state estimation with evolving networks and Monte Carlo methods. IEEE Trans. Signal Processing **59**(4) (2011) 1383–1396
12. Hernandez-Leal, P., Gonzalez, J.A., Morales, E.F., Sucar, L.E.: Learning temporal nodes bayesian networks. Int. J. Approx. Reasoning **54**(8) (2013) 956–977
13. Ivanov, Y.A., Bobick, A.F.: Recognition of visual activities and interactions by stochastic parsing. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8) (2000) 852–872
14. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley (1990)
15. Lee, K., Kim, T.K., Demiris, Y.: Learning action symbols for hierarchical grammar induction. In: ICPR. (2012) to appear
16. Li, K., Hu, J., Fu, Y.: Modeling complex temporal composition of actionlets for activity prediction. In: ECCV. (2012) 286–299
17. Manfredotti, C.E., Fleet, D.J., Hamilton, H.J., Zilles, S.: Simultaneous tracking and activity recognition. In: ICTAI. (2011) 189–196
18. Nguyen, N.T., Phung, D.Q., Venkatesh, S., Bui, H.H.: Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models. In: CVPR. (2005) 955–960
19. Niebles, J.C., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. Int. J. Comput. Vision **79**(3) (2008) 299–318
20. Oh, S.M., Rehg, J.M., Balch, T.R., Dellaert, F.: Data-driven MCMC for learning and inference in switching linear dynamic systems. In: AAAI. (2005) 944–949
21. Ryoo, M.S., Aggarwal, J.K.: Stochastic representation and recognition of high-level group activities. Int. J. Comput. Vision **93**(2) (2011) 183–200
22. Ryoo, M.S., Aggarwal, J.K.: Semantic representation and recognition of continued and recursive human activities. Int. J. Comput. Vision **82**(1) (2009) 1–24